WL-TR-93-1086

APPLICATION OF THE NETWORK DISCRIMINANT FUNCTION TO LEARNING

Mr. D. Wicker

Jun 93

Interim Report for Period January 1993 - June 1993

Approved for public release; Distribution is Unlimited

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIELCOMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-7409

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Devert W. Wicker

DEVERT W. WICKER
Electronics Engineer
System Concepts Section

John E Jacobs

JOHN E. JACOBS
Chief, System Concepts Section
Applications Branch

William E Moore

WILLIAM E. MOORE
Acting Chief, Applications Branch
Mission Avionics Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/AART-2, WPAFB, OH 45433-7408 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Jun 93	Interim Jan 93 - Jun 93

4. TITLE AND SUBTITLE

Application of the Network Discriminant Function to Learning

5. FUNDING NUMBERS

PE 63203F
PR 69DF
TA 00
WU 00

6. AUTHOR(S)

Devert Wicker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Avionics Directorate
Wright Laboratory
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7409

8. PERFORMING ORGANIZATION REPORT NUMBER

WL-TR-93-1086

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Avionics Directorate
Wright Laboratory
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7409

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

WL-TR-93-1086

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This introduces methods of applying the Network Discriminant Function (NDF) to training and automatic architecture selection of feed-forward multiple layer perceptrons (MLPs) used for Pattern Recognition (PR). Knowledge of the NDF may be used to impact MLP architecture choices. The NDF gives a new insight into MLP performance in that it reflects how well the MLP is clustering the data. If clustering is an intrinsic capability of the MLP which makes it perform well, then the NDF should correlate with other performance measures such as the Sum Squared Error. Architecture choices and training technique need to be selected as to aid the clustering ability of the MLP. The major goals of the research are to establish the NDF as a useful criteria and to compare it to other criteria such as Cascade-Correlation's error correlation. The NDF-Cascade (NDFC) learning architecture and the MLP NDF are proposed, tested and evaluated.

14. SUBJECT TERMS

Neural Networks, Pattern Recognition, Network Discriminant Function, Machine Learning, Training, Architecture Selection Cascade-Correlation

15. NUMBER OF PAGES

36

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Foreword

The work presented here is also being performed as dissertatation research under the guidance of Dr. Kevin Kirby, Professor of Computer Science and Engineering at Wright State University in Fairborn, OH. The background research was initially performed under the direction of Dr. Alistar McAulay who formerly was of Wright State University but now is the Chairman of the Department of Electrical Engineering and Computer Science at Lehigh University in Bethlehem, PA. The author wishes to thank them for their help and guidance.

Contents

List of Figures

List of Tables

1 Introduction

This introduces methods of applying the Network Discriminant Function (NDF) to training and automatic architecture selection of feed-forward multiple layer perceptrons (MLPs) used for Pattern Recognition(PR). Knowledge of the NDF and of the MLP clustering ability may be used to impact MLP architecture choices as well as give researchers an important new handle on how to evaluate MLP performance. In the past, the sum of the square error (SSE) has been the main metric that researchers had to consider. The NDF gives a new insight into MLP performance in that it reflects how well the MLP is clustering the data. If clustering is an intrinsic capability of the MLP which makes it perform well then the NDF should correlate with other performance measures such as the SSE. Architecture choices and training technique need to be selected so as to aid the MLP clustering ability. The impact of architecture choices based on the NDF is an area which has not yet received attention in the literature.

The major goals of the research are to establish the NDF as a criteria useful for architecture selection and to relate and compare the NDF to other criteria such Cascade-Correlation's error correlation. The Background section discusses MLPs and pattern recognition and in particular discusses 1 out of N networks, linear last layer networks, Cascade-Correlation(CC) networks, and functional-link networks. Next, application of the NDF is discussed. The NDF-Cascade(NDFC) learning architecture and the MLP NDF are proposed. A series of tests are described, test results are given, and each test is discussed individually. Then, the results are discussed and new suggestions for research made. Finally, the research is summarized.

2 Background

PR is a well established field of study with a rich body of techniques and theories. The amount of research on artificial neural networks (ANNs) has grown in the past several years. This is evident from the creation of new research journals such as the IEEE Transaction on Neural Networks and Neural Networks. The areas of sensor processing, control, and data analysis have successfully applied neurocomputing to hard problems [1]. Perhaps the most significant success has occurred within the area of PR. The relationship of ANNs and PR systems is synergistic [2]. PR gives theoretical basis for functions that need to be performed while ANNs address many of the hardest problems that PR systems have in practice. ANNs do not solve all the problems of PR systems but they are improving PR capabilities. The lack of understanding of exactly what mapping ANNs are performing will slow their acceptance. This is especially true for critical process situations. Engineers need to know that the PR system is performing logically and may want to modify what it is

doing. Automatic selection of architecture requirements such as number of layers, number of nodes in a particular layer, and type of node functions is important to allow for more flexible systems and to expand usage to lesser skilled users. These goals require greater understanding of the MLPs and how the architecture impacts their performance.

2.1 Dividing the Decision Space

MLPs can divide the decision space [3]. A single node provides a boundary in space. A linear node provides a linear decision boundary while a nonlinear node provides a nonlinear decision boundary. Adding additional linear layers provides no benefit as the weight matrices can simply be multiplied to provide an equivalent single layer network. This points out an inherent limitation of linear MLPs, they only work well on linearly separable problems. Nonlinear nodes make the addition of more layers useful. The second layer combines the decision boundaries of the first layer to provide decision regions while the third layer combines the regions into areas. In this way the nonlinear MLP allows the decision space to be further divided based on decision boundaries created in the first layer. A classic example is the exclusive-or problem which is not linearly separable yet a three layer MLP can successively learn it. This added ability of nonlinear MLPs may be attributed to the fact that unlike linear MLPs they can increase the rank of the data [4]. Webb and Lowe experimentally demonstrated this by showing improved class separation of a nonlinear hidden layer with higher-dimensional space. They suggest that this is because nonlinear MLPs can increase rank into a space where discrimination is easier to perform. As data rank is a linear measure, a hard and strict rule for determining layer widths as a function of data rank seems unlikely. Given that the MLP can divide the decision space, the next step is to derive desirable decision boundaries.

The MLPs considered here are trained so as to minimize the sum of square error. Let $\vec{t}_p$ and $\vec{o}_p \in R^n$ be the pth desired target pattern vector and actual output vector, respectively. Assume that each node has a weighted connection to a node with fixed output 1. This allows bias thresholds to be treated implicitly. As the input vectors are given, $\vec{o}_p$ is a function of only the interconnection weights. The sum square error (SSE), E, over a given training set may then be expressed as:

$$E = \sum_{p=1}^{P} d_p \parallel \vec{t}_p - \vec{o}_p \parallel^2 \tag{1}$$

d_p is a constant which weights the error for pattern p. $\vec{o}_p$ is a function of the weights so E is also a function of the weights which may be minimized by adjusting the weights. This is the well known least squares or autoregression problem. One interpretation this gives MLPs is that they are simply mapping functions which are curve fitted to a set of data points, i.e., the training set pairs.

2.2 The 1 out of N MLP

Of particular interest to this research are the subset of MLPs which use 1 out of N encoding which is useful for multi-class problems. Each output corresponds to particular class. A 1 represents the input vector belongs to the class while a 0 implies it does not. Given mutually exclusive classes only 1 out of the N output nodes should be 1 while all others are 0. The outputs of 1 out of N encoded MLPs have proved to approximate the optimal Bayes discriminant functions [5]. Each output is a minimum mean squared-error estimate of the corresponding class discriminant function which is the probability of the class given the input. The proof applies to any technique which attempts to minimize the mean squared error and the desired outputs are 0 and 1. It does not depend on what type of nodes are used or the architecture of the network. Each output represents the *a posteriori* probability for the corresponding class. Knowing that the output is the *a posteriori* probability allows selection of sensible decision thresholds on the outputs of MLPs. The quality of the approximation does depend on a good sampling of the data and an architecture capable of accurate approximation of the *a posteriori* probabilities. Importantly, nonlinear MLPs can perform any mapping [6] so any probability distribution can be modeled by MLPs.

2.3 Linear Last Layer MLPs

Other authors have investigated 1 out of N encoded MLPs which use linear nodes in the output layer [4, 7, 8, 9, 10, 11]. For linear MLPs, E is known to have a unique minimum corresponding to the projection onto the subspace generated by the first principal vectors of a covariance matrix associated with the training patterns [9]. Gallinari[7] considers a linear case with two linear layers. The first layer was proven mathematically to perform a discriminant analysis projection while the second layer performs a classification on the outputs of the hidden units. For MLPs with multiple nonlinear layers ending with a linear layer, the MLP has been experimentally demonstrated to map data into smaller and smaller clusters which are easier to separate as the data maps from layer to layer. Webb[4] proves that the outputs of the hidden layer into the last linear layer can be shown to maximize the NDF:

$$J_N = Tr\{S_B S_T^+\} \tag{2}$$

where:

$$S_T = \sum_{p=1}^{P} (\vec{h}_p - \vec{m}_H)(\vec{h}_p - \vec{m}_H)^t \tag{3}$$

$$S_B = \sum_{k=1}^{N} n_k^2 (\vec{m}_{H_k} - \vec{m}_H)(\vec{m}_{H_k} - \vec{m}_H)^t \tag{4}$$

where N is the number of classes, P is the number of training patterns, n_k is the number of patterns in class k, $\vec{h}_p$ is the outputs of the last hidden layer for pattern p, $\vec{m}_H$ is the mean output vector at the hidden units over all patterns, and $\vec{m}_{H_k}$ is the mean output vector at the hidden units for class k patterns. S_B is similar to the between-class scatter matrix of Duda and Hart[12] while S_T^+ is the pseudo inverse of the total unnormalized class covariance which is also known as the total scatter matrix. Weights are chosen so as to maximize the NDF in the space spanned by the outputs of the final hidden layer. The part of the network up through the output of the hidden layer clusters the classes based on the NDF while the final layer performs an optimal mapping onto the targets. So the MLP optimizes a specific feature extraction and performs classification simultaneously.

2.4 NDF Adjustment

Lowe[8] discusses pattern error weighting (choosing d_p) and target coding. Weights and target assignment values can be selected so as to directly compensate for uneven class distributions. The cost of assigning a class incorrectly can also be incorporated by selection of target vector (i.e., output is something other than just 1s and 0s) but requires a post process decision on the output vector. The way the MLPs with a linear output layer perform is understood well enough to allow modification of the discriminant functions. In particular, the ability to include the cost of wrong classification is important. For example, identifying a friendly fighter as an enemy fighter could lead to unnecessary fratricide so this cost consideration may be considered in network design. Output target vector values and pattern error weights can be selected logically based on the selection of a desirable NDF.

2.5 Fisher's Discriminant

A well known discriminant used in PR is Fisher's discriminant[12, 13]:

$$J_F = \frac{|\, S_B \,|}{|\, S_W \,|} \tag{5}$$

where S_B is the standard between-class scatter matrix: covariance:

$$S_B = \sum_{k=1}^{N} n_k (\vec{m}_{H_k} - \vec{m}_H)(\vec{m}_{H_k} - \vec{m}_H)^t \tag{6}$$

S_W is the within-class scatter matrix:

$$S_W = S_T - S_B \tag{7}$$

The determinant is a measure of volume spanned by the points, so increasing the determinant corresponds to increasing the volume and, thus, the distance between points. Thus,

maximizing Fisher's discriminant corresponds to increasing $| S_B |$, the distance between classes, while decreasing $| S_W |$, the distance between points in a class. The NDF S_B is similar to Fisher's S_B but it weights the larger classes more heavily.

2.6 Cascade-Correlation and Functional-Link

For a more complete description of Cascade-Correlation (CC), see Fahlman's report[14]. CC is a relatively new architecture generating algorithm which uses error correlation to build up to the final network. The learning process begins with two layers. The first layer consists of the input and the bias node while the other is simply the output layer. The network is trained. If the network doesn't work then a new unit is created which takes input from all other nodes except for the output layer nodes. Thus, the new unit becomes a new layer in the network. The new unit is trained to maximize its output correlation to the error:

$$S = \sum_{o=1}^{N} | \sum_{p=1}^{P} (V_p - \bar{V})(E_{p,o} - \bar{E}_o) | \tag{8}$$

where V_p and $E_{p,o}$ are respectively the candidate unit's output value for pattern p and the current network error for pattern p at output o, and where $\bar{V}$ and $\bar{E}_o$ are respectively the average of the candidate unit's output and output o's error. Thus, the unit becomes a feature detector for problematic patterns. Once the new unit is added, the weights into the output layer are retrained and the learning process repeats until the network works successfully. Instead of only one unit, a pool of units with the same or varying functions can be used. The unit with the maximum error correlation is selected. This increases the odds of getting a good unit.

The cascade architecture generated by CC can be considered to be an instantiation of the functional-link net discussed by Pao[15]. In the functional-link concept only two layers are ever used. The first layer consists of the inputs and functions on the inputs. Pao showed that the right set of functions leads to networks that train easily to good solutions. CC simply uses error correlation as a basis for selecting new functions of the original inputs to be added to the first layer.

3 Applying the NDF

Does the NDF have properties which makes it a good search criteria for functions or aiding weight learning? For linear last layer MLPs, SSE is maximized when the NDF is. The NDF appears to be the measure for how MLPs clusters data as it maps through the layers. Intuitively, the NDF measures how separable the data is. A bigger NDF is better as the SSE learning seeks to maximize it. Therefore, the NDF is a basic property which may be

useful in training and automatic architecture selection. For example, some units and their functions may aid the clustering process better than others and, thus, lead to MLPs which train quicker as well as perform better. The NDF may be useful for selecting these units from a pool of units or actually growing new units as in CC. The NDF may be useful as another criteria in the gradient search cost function. These ideas are applied in the following algorithm proposals.

3.1 Unit Selection

In the functional link concept presentation by Pao[15], units were simply the possible combinations of the original inputs going into "and" gates. These units could be put into a pool of units where an algorithm could consider each for selection. In this case, the NDF could be used. Initially, only the original inputs are given. For each unit in the pool, calculate its outputs for each pattern. Train the net and check if the solution is good. If not, select a unit from the pool. For each unit, calculate the NDF assuming its output is added to the current input pattern. Remove the unit with the maximum NDF from the pool and add it permanently to the pattern. Repeat the training and selection process until the network gives a good solution or the pool is emptied.

3.2 The NDF-Cascade Learning Architecture

The NDFC approach is similar to CC. Instead of choosing units which maximize error correlation, units are selected to maximize the NDF. Based on the requirements leading to the NDF, the last layer consists of linear nodes and 1 out of N encoding is used. As the last layer is linear, weights into the output layer can be directly calculated using standard linear least squares parameter calculations[16].

3.3 The NDF MLP

The NDF MLP approach selects MLP weights using the cost:

$$C = E - KJ_N \tag{9}$$

where K is the NDF cost weight. As minimizing SSE implies maximizing the NDF, the global minimum should be the same but the derivative information may be different and, thus, lead to learning good solutions that minimizing SSE by itself would not find.

4 Tests and Results

A number of tests were performed which explore the NDF and its usefulness. For test purposes, a network was deemed to have a good result when for each pattern, the correct

Unit Number	Unit Function
1	$i_0 \cap i_1$
2	$i_0 \cap i_2$
3	$i_2 \cap i_2$
4	$i_0 \cap i_1 \cap i_2$

Table 1: Unit Numbers for 3-Bit Problems

class output is maximum.

4.1 NDF vs. Fisher

To test the correlation between NDF and Fisher, a two-input three-class problem was constructed. Normal distributions were used to generate the inputs. Both inputs for each class used the same normal distribution. Letting $N(\mu, \sigma)$ represent a normal distribution with mean, μ, and standard deviation, σ, the distributions used were $N(0,1)$, $N(3,1)$, and $N(-3,1)$ for classes one, two, and three. Given 1000 data sets each with 100 points randomly selected to be from one of the three classes and then randomly selected from that class, the correlation coefficient between NDF and Fisher was found to be 0.278. If the correlation coefficient between the NDF and Fisher were one, then NDF and Fisher could be considered equivalent.

4.2 Unit Selection Tests

Analysis on the 3-bit parity problem and the 3-bit symmetry problem has been performed in order to demonstrate the feasibility of using the NDF to select units from a pool of units available to a functional link. The algorithm considered simply selects from the unit pool the unit which maximizes the criteria. For comparison purposes CC S has also been considered. For the test problems, the units in the unit pool are numbered as in Table 1 where i_j is input j.

4.2.1 3-Bit Parity

For the 3-bit parity problem, Table 2 shows all the NDF values for permutations used in the possible search paths for determining which units to add. Unit 4 would be the first unit added because it gives the maximum NDF of 1.00 while adding any other unit gives a zero NDF. Given Unit 4, adding any of the remaining units gives an NDF of 1.33 so any of these could be used. The next unit selected results in an NDF of 2.00 and finally adding all the units gives an NDF of 4.00. At this point, a good solution is obtained.

Used Units	Possible Unit	NDF Value
	1	0.00
	2	0.00
	3	0.00
	4	1.00
4	1	1.33
	2	1.33
	3	1.33
4,1	2	2.00
	3	2.00
4,2	1	2.00
	3	2.00
4,3	1	2.00
	2	2.00
4,1,2	3	4.00
4,1,3	2	4.00
4,2,3	1	4.00

Table 2: 3-Bit Parity Unit Selection by NDF

Used Units	Possible Unit	S Value
	1	0.3750
	2	0.6250
	3	0.6250
	4	0.6875
4	1	0.2500
	2	1.0000
	3	1.0000
4,2	1	0.5000
	3	1.1666
4,3	1	0.5000
	2	1.1666
4,2,3	1	0.5000

Table 3: 3-Bit Parity Unit Selection by S

Table 3 shows all the S values for permutations used in the search paths for determining which units to add. S order of selecting the units turned out to be Unit 4, Unit 2 or 3, remaining of Unit 2 or 3 not picked and finally Unit 1.

4.2.2 3-Bit Symmetry

For the 3-bit symmetry problem, Table 4 shows all the permutations for use of units and their corresponding NDF value. Based on the NDF table, Unit 2 would be used first. The network can learn the problem given only this node. Given Unit 2 adding any of the other units does not increase the NDF.

Used Units	Possible Unit	NDF Value
	1	0.00
	2	4.00
	3	0.00
	4	0.00
2	1	4.00
	3	4.00
	4	4.00

Table 4: 3-Bit Symmetry Unit Selection by NDF

Used Units	Possible Unit	S Value
	1	0.3750
	2	2.3750
	3	0.3750
	4	1.1875
2	1	0.5000
	3	0.5000
	4	0.2500

Table 5: 3-Bit Symmetry Unit Selection by S

Table 5 shows that S also directly selects Unit 2.

4.2.3 Discussion of Unit Selection Tests

The unit selection tests show that the NDF can be used as a criteria for selecting units to be used in a functional link. The NDF selects units which lead to good answers. CC S criteria appears to select units in a more restrictive fashion than the NDF criteria but it selects units which also maximize the NDF. In the 3-bit symmetry problem, the NDF stops growing once the only needed unit is added. This may be an indicator that further node selection is worthless. CC S does not have a similar heuristic at this point in the learning.

4.3 NDFC Tests

The NDFC approach has been implemented. Problems were run in order to test NDFC and allow comparison of it to CC. In both approaches, conjugate gradient[16] is used to train the new units while the weights into the linear output layer are calculated using the singular value decomposition approach to solve linear least square problems[16]. The cascade units used the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{10}$$

Derivatives are calculated using numerical methods rather than by direct calculation of formula.

4.3.1 3-Bit Parity Test

Both methods were tested on the 3-bit parity problem. First, a pool size of 1 was used. CC required 2 cascade units to learn the data. The NDF started at 0.0 with no new units, increased to 1.0 with 1 unit, and finally increased to 4.0 with 2 units. The number of

incorrectly learned patterns started at 4, dropped to 1, and finally ended at 0. NDFC showed no ability to learn as the error count stayed at 4 after 3 cascade units were added and the NDF remained a constant 0.0. Next, the pool size was increased to 10 where each unit starts with different random weights. Both NDFC and CC required only a single new unit to learn the 3-bit parity problem. For both methods, the number of incorrectly learned patterns started at 4 and dropped to 0 with the new unit. For both methods, the NDF starts at 0.0 with no new units and increases to 4.0 with the new unit.

4.3.2 4-Bit Parity Test

Both methods were tested on the 4-bit parity problem. Each method used a pool size of 10. CC required 2 new units to learn the 4-bit parity problem. The number of incorrectly learned patterns started at 8, dropped to 4, and finally ended at 0.0. The NDF started at 0.0, increased to 1.6, and finally ended at 8.0. NDFC required 3 units. The number of incorrectly learned patterns started at 8, dropped to 1, stayed at 1, and finally ended at 0. The NDF started at 0.0, increased to 3.79, stayed at 3.79, and finally ended at 8.0. Cascade Unit 2 had 0.0 output variance which means the unit does nothing so a check was added to throw out any candidate units with variance less than 0.001. With this change, the second unit took the NDF to 8.0 and reduced the number of incorrectly learned patterns to 0.

4.3.3 Quadrant Test

Both methods were tested on a contrived two input data set. Class 1 data points were points which had both inputs in the range 0 to 5. Otherwise they were Class 2. This was sampled using a grid of points, 11 across each axis in the range 0 to 10. Thus Class 1 forms the lower left quadrant of the samples. Using a pool size of 10, CC required 6 cascade units to obtain the answer while NDFC only required 1. Figures 1 and 2 show the mappings obtained by NDFC as learning progressed while Figures 3, 4, and 5 show those for CC. These figures are maps of the classification obtained for a sample grid with each axis having 64 points ranging from -1 to 11. To verify this performance, the test was repeated using Fahlman's own CC code, "cascor." Given 20 trials with a pool size of 8, the smallest network had 4 cascade units. Given 10 trials with a pool size of 64, the smallest network again had 4 cascade units. Given 20 trials, a pool size of 8, and a sigmoid last layer rather than linear, the smallest network had 2 cascade units.

4.3.4 Iris Test

Both methods were tested on Fisher's iris plant data[12, 17, 18]. This is perhaps the best known database found in the pattern recognition literature. Each sample has the four predictive attributes of sepal length, sepal width, petal length, and petal width for three

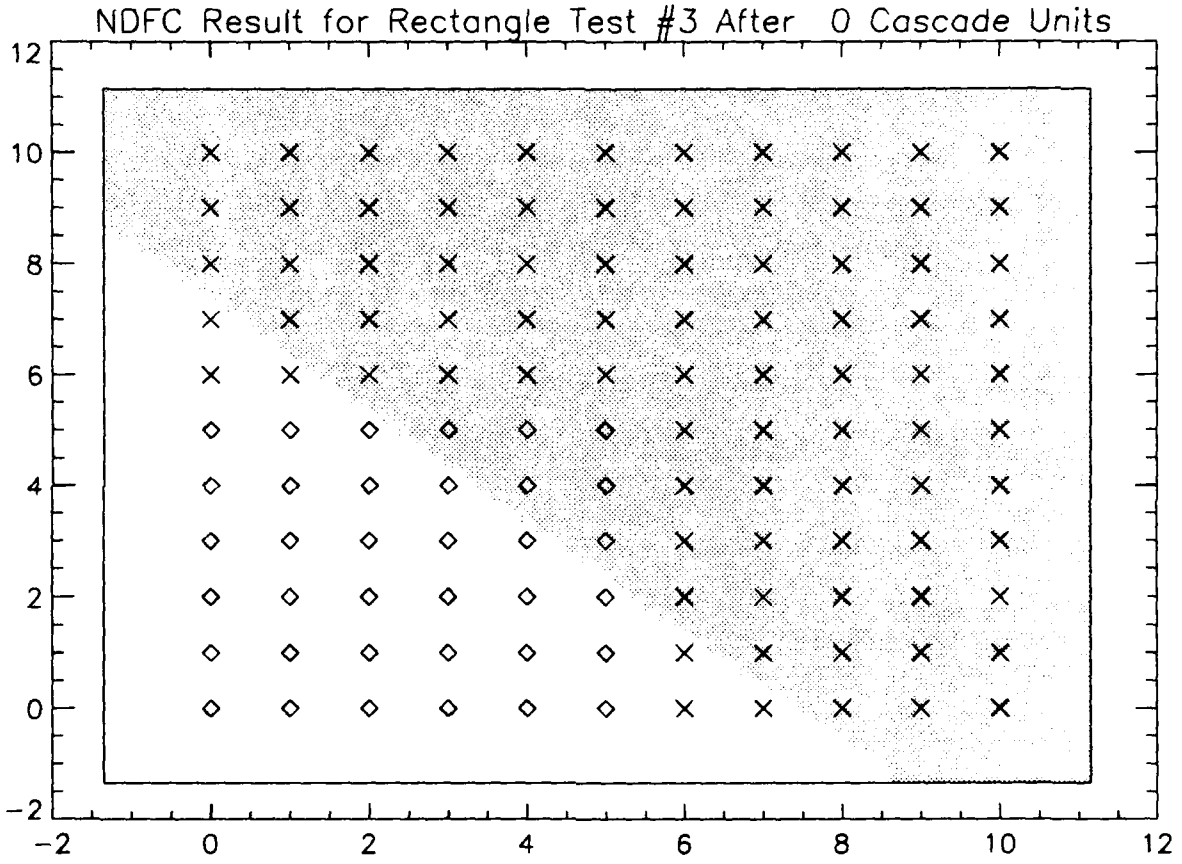

Figure 1: NDFC Resulting Map for the Quadrant Test with No Cascade Unit.

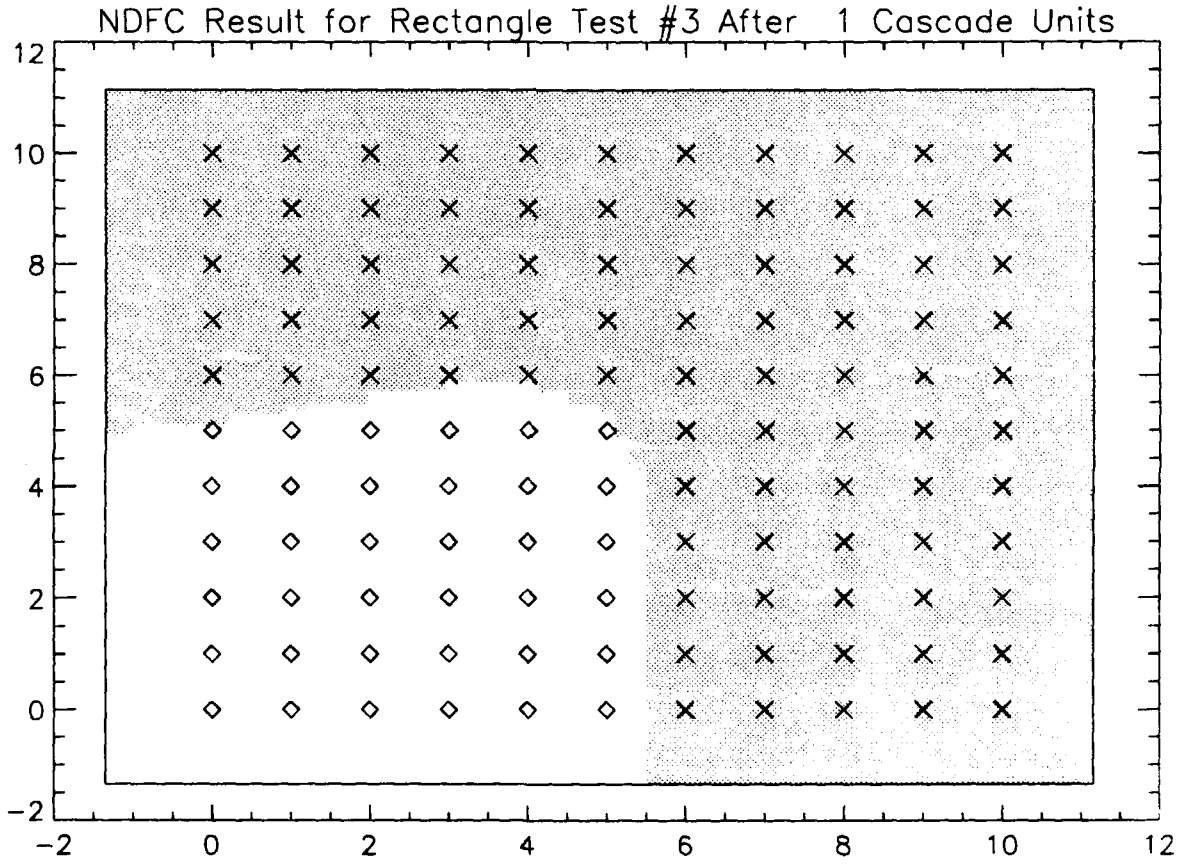

Figure 2: NDFC Resulting Map for the Quadrant Test with One Cascade Unit.

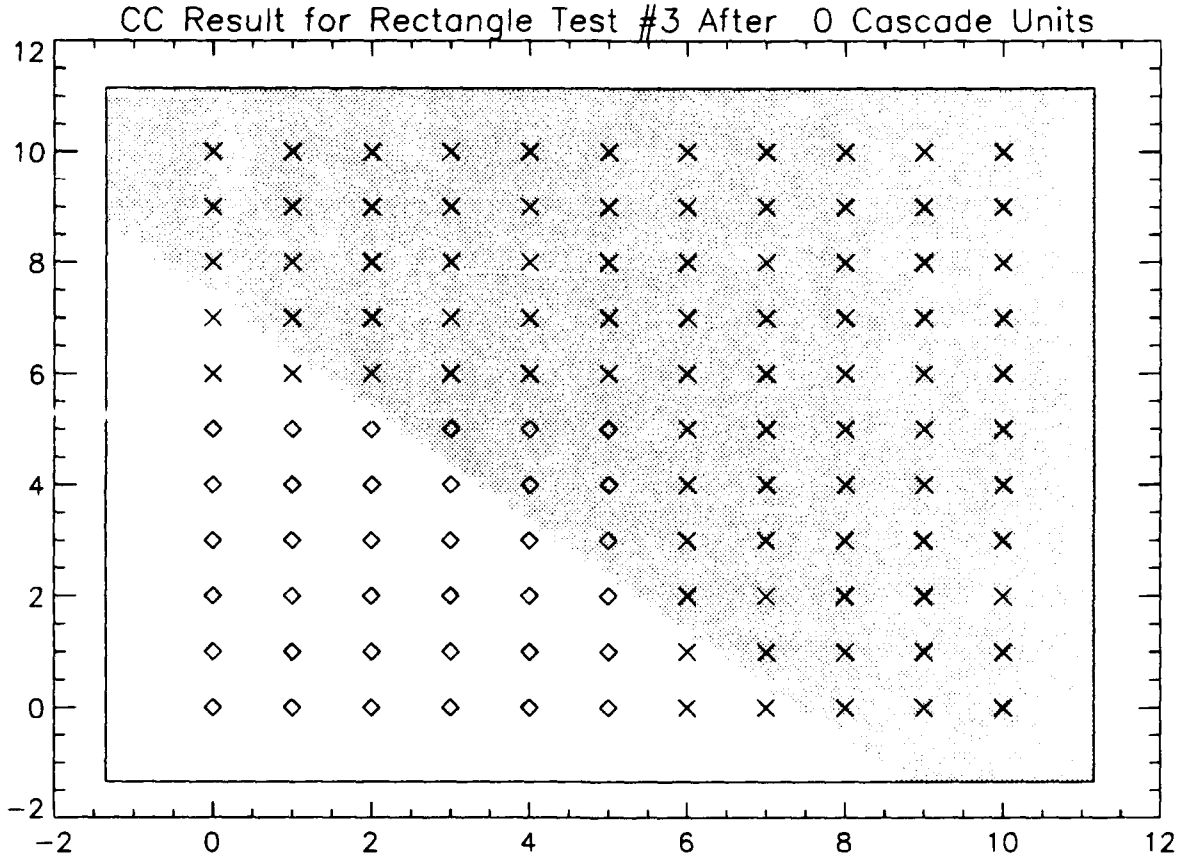

Figure 3: CC Resulting Map for the Quadrant Test with No Cascade Unit.

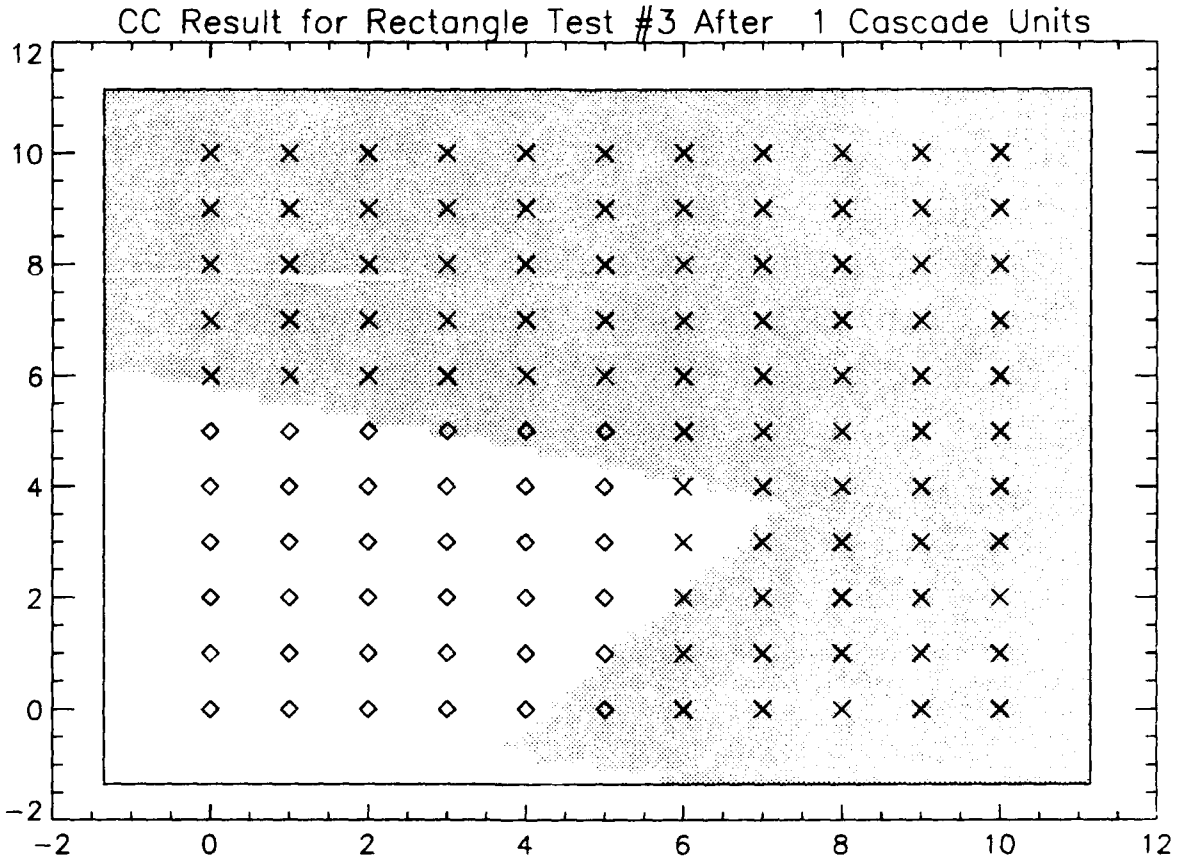

Figure 4: CC Resulting Map for the Quadrant Test with One Cascade Unit.

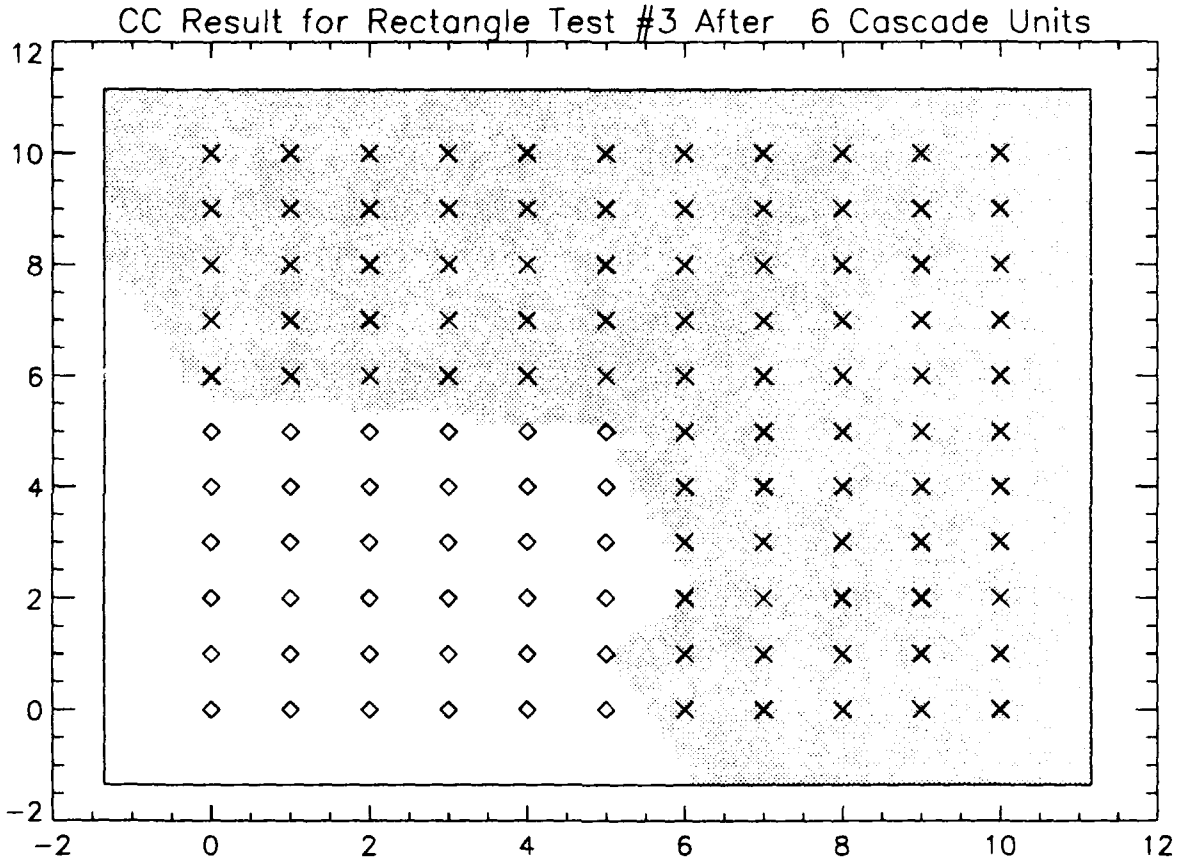

Figure 5: CC Resulting Map for the Quadrant Test with Six Cascade Units.

species of iris which are Setosa, Versicolor, and Viginica. The data set has 50 samples of each class for a total of 150 samples. The Setosa is linearly separable from the other two classes while the Versicolor and Virginica are not linearly separable from each other. First, a pool size of 10 was used. Figure 6 shows the effect of each cascade unit on the error count using NDF and CC to choose units. Both methods required five new units to learn the data correctly. Figure 7 shows the NDF for both methods as the new units are added. Next, each method was given a pool size of 50. For NDFC, two different candidate unit variance limit checks, 10^{-20} (Zero) and 0.001 were tried. Figures 8 and 9 show the results. CC and NDFC with variance limit 10^{-20} failed to learn the data after 5 units while NDFC with variance limit 0.001 again required 5 units to learn the data.

4.3.5 Criteria Derivative Tests

To test whether or not NDFC and CC agree on units created, criteria derivative tests were performed using the 3-bit parity data. Using the initial data, candidate units were trained based on the NDFC and CC S. Each method was given 10 tries to maximize its criteria. The candidate units created were then given as initial conditions to the opposing method. In both cases, the initial cost derivative was found to be zero so no changes were made.

4.3.6 Discussion of NDFC Tests

The NDFC test results demonstrate that NDFC can be used to create cascade networks which reach good solutions. The pool size impacts the performance of both algorithms. In the 3-bit parity test, increasing the pool size helped both algorithms find a solution requiring only one unit where before CC required two units and NDFC did not work at all. NDFC found a smaller network in the quadrant case. In the case of the iris data, CC performed oddly in that a pool size of 10 took only 5 units while a pool size of 50 still did not give a good solution after 9 units. As the larger pool size case included all the same starting weights as the smaller pool size, this pool size effect suggests that CC may be a greedy algorithm. The success of CC and NDFC with a pool size of 10 showed that a good solution given 5 units exists yet CC failed to find one even given a pool size of 50. Putting a lower limit on the candidate unit output variance proved to be useful for the 4-bit parity and the iris test in that it increased NDFC performance. Choice of the limit needs further study. In particular, knowing a good value for the limit would avoid useless runs. The criteria derivative tests show that the NDF and CC S share common zero derivative points. This suggests that there is some agreement between them as to which weights to select.

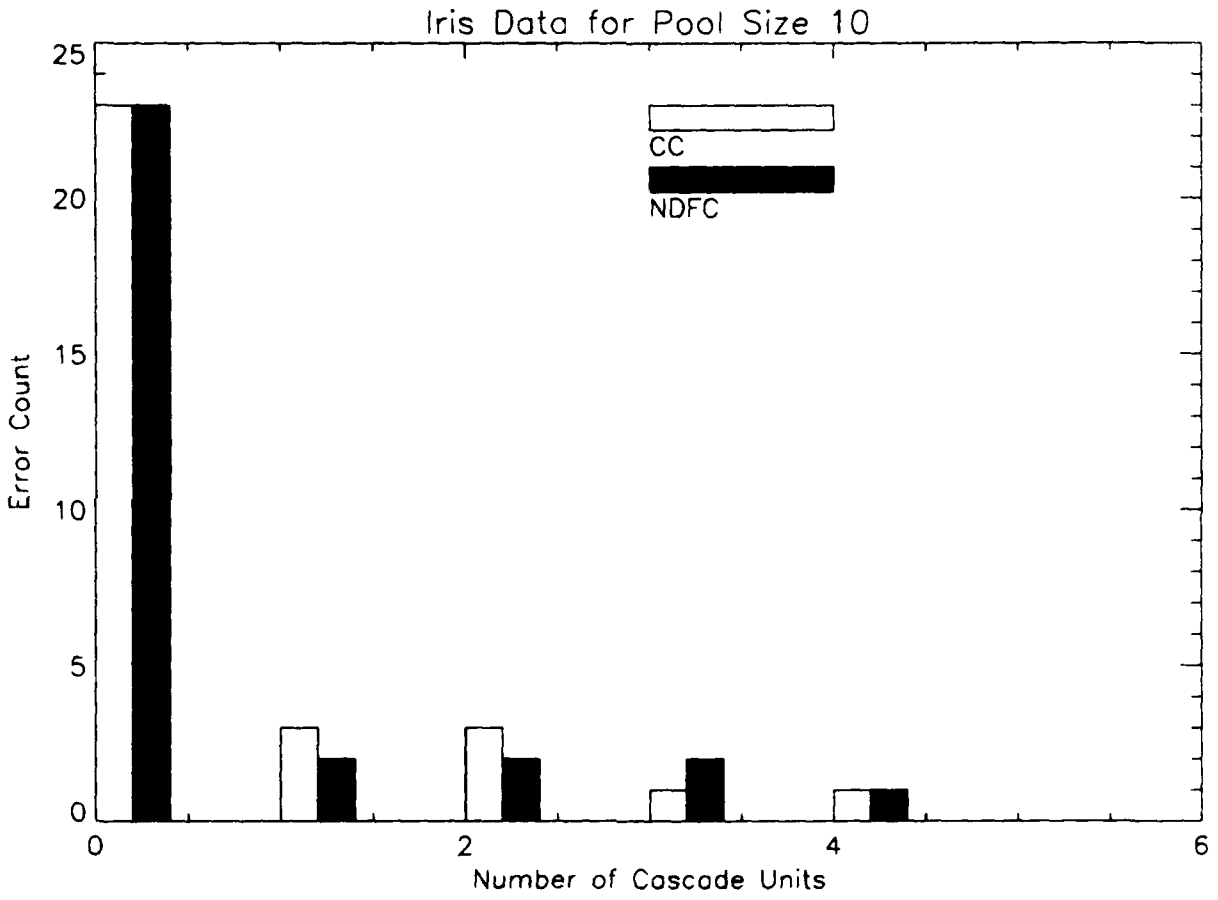

Figure 6: Number of Cascade Units vs. Error Count (Pool Size 10)

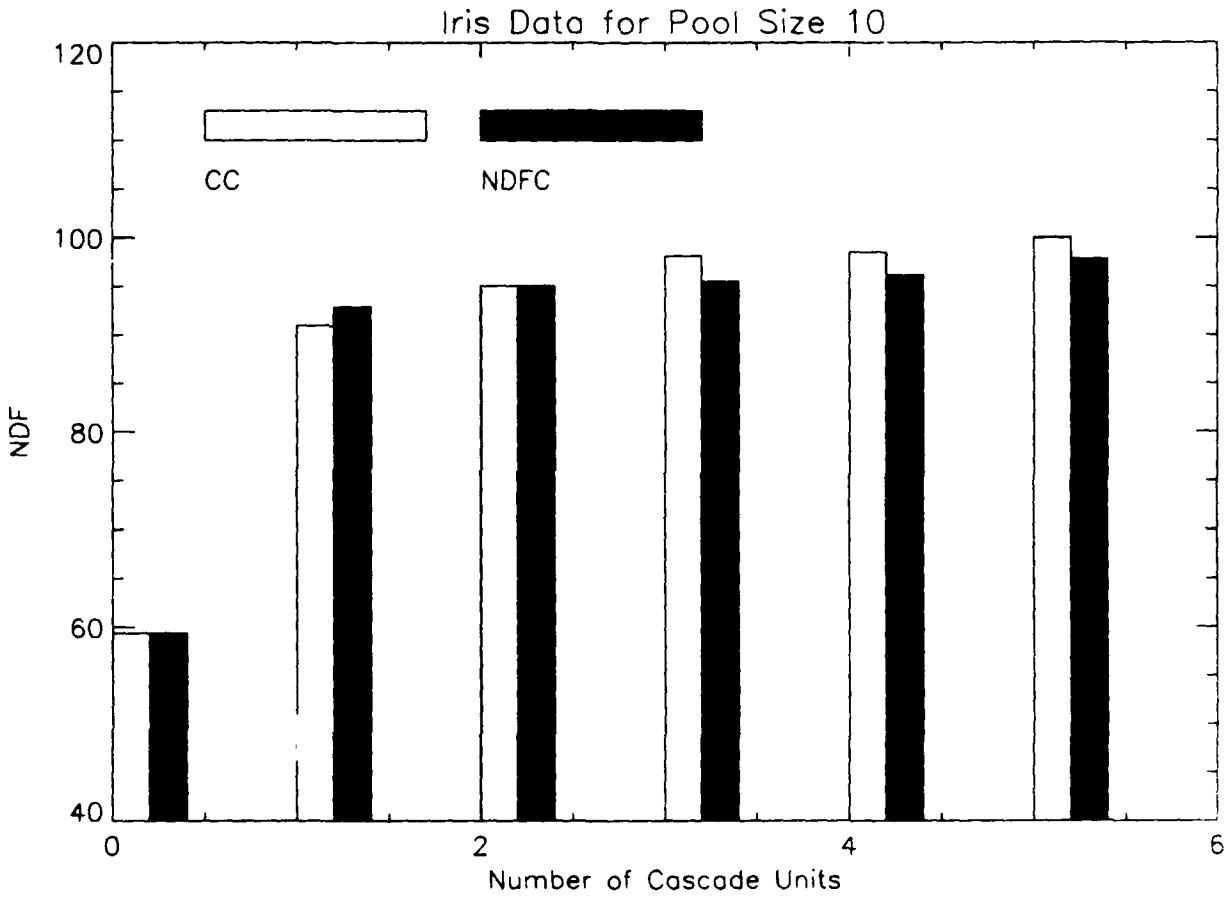

Figure 7: Number of Cascade Units vs. NDF (Pool Size 10)

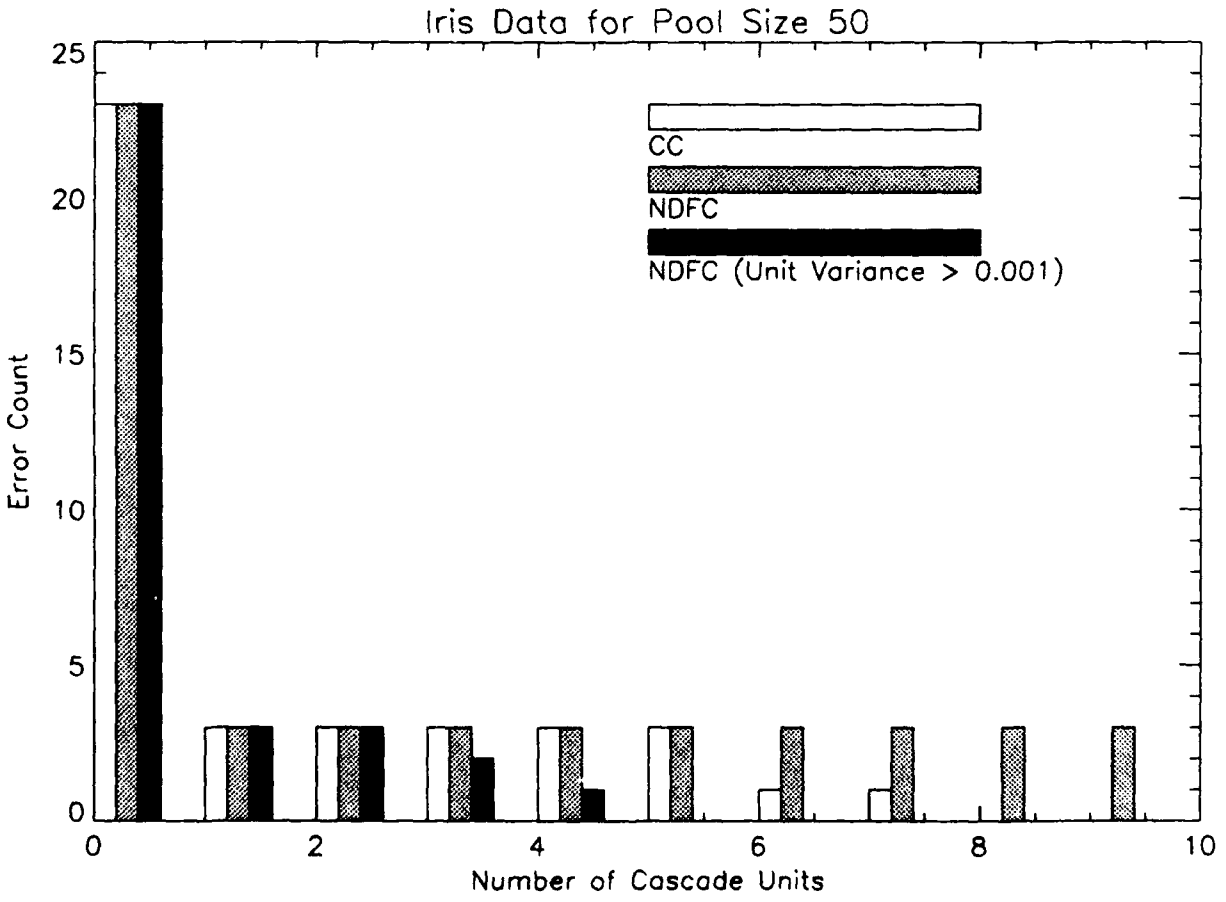

Figure 8: Number of Cascade Units vs. Error Count (Pool Size 50)

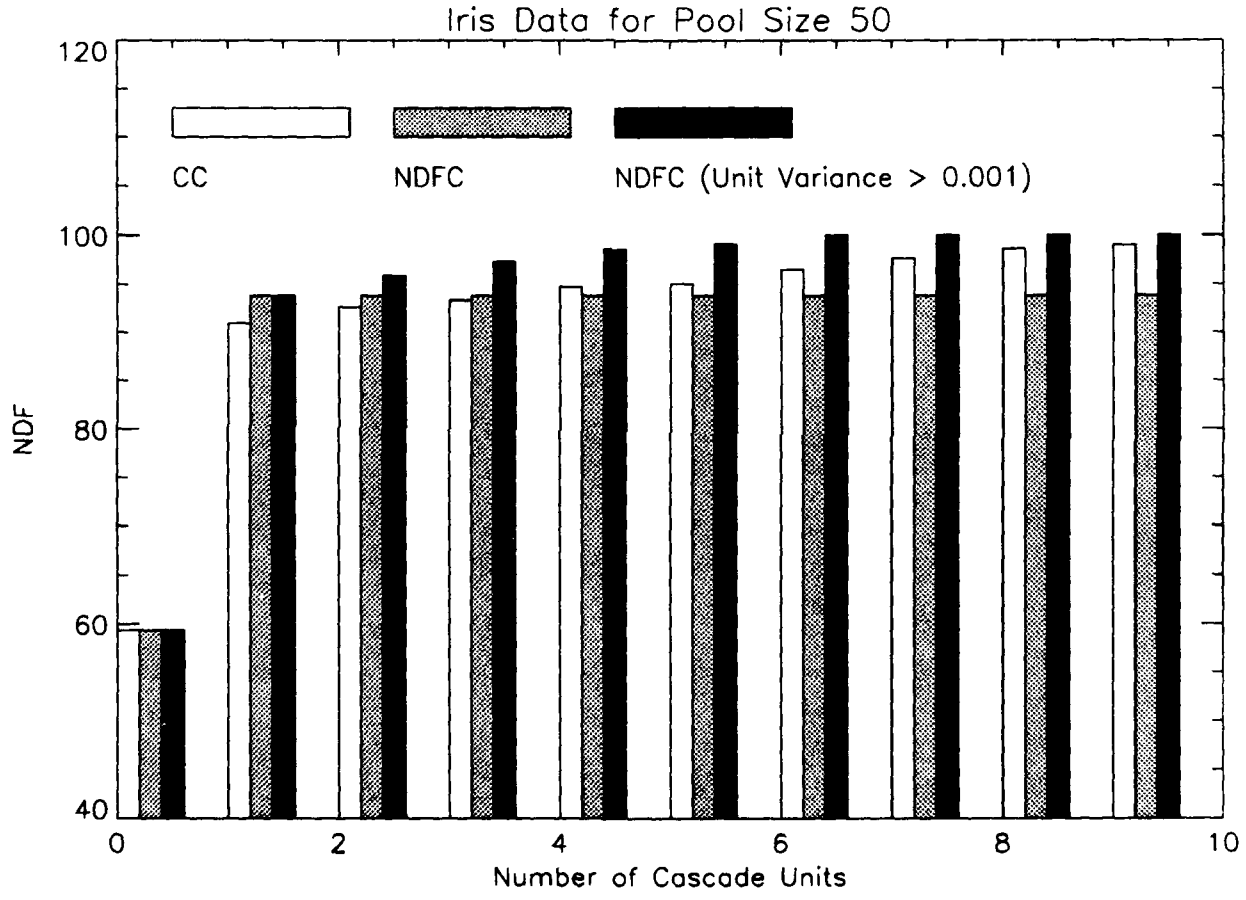

Figure 9: Number of Cascade Units vs. NDF (Pool Size 50)

NDF Weight	2-bit Symmetry	4-bit Symmetry
	(2 2 2)	(4 3 2)
0.0	1	1
1.0	7	7
8.0	18	45
20.0	24	63
100.0	25	41

Table 6: Impact of NDF on MLP Learning. Number of Good Solutions when Initial Weights are $U(-0.2, 0.2)$

NDF Weight	2-bit Symmetry
	(2 2 2)
0.0	157
1.0	23
8.0	58
20.0	48

Table 7: Impact of NDF on MLP Learning. Number of Good Solutions when Initial Weights are $U(-0.6, 0.6)$

4.4 NDF MLP Tests

To see if adding the NDF into the cost used to optimize an MLP helps the learning, tests were run using different values of the NDF cost weight, K. Conjugate gradient was used to train the MLPs. Hidden layer nodes were sigmoid function units while the last layer used linear nodes. Weights were random selected based on a uniform distribution of values between plus or minus 0.2 ($U(-0.2, 0.2)$). Given the same 500 sets of initial random weights, the number of good solutions were counted. Table 6 shows the results for a couple of problems. The MLP dimensions used for each problem are given in parenthesis, e.g., (4 3 2) means that the 4-bit symmetry problem used a net with a 4-node-input layer, a 3-node-hidden layer, and a 2-node-output layer. Table 7 shows the impact of increasing the weight distribution range to $U(-0.6, 0.6)$ on the 2-bit symmetry problem.

4.5 Discussion of NDF MLP Tests

The tests show that using the NDF in the cost increases the odds of getting a good solution only if poor initial conditions are used. In fact, using it with good initial weights appears

to be detrimental. This makes it of limited value given the calculation overhead and the fact that it is easy to adjust the initial weight range. If finding good initial weights proves hard, then using the NDF could be useful.

5 Discussion

The NDF is useful for training functional link networks including cascade networks. The success of the unit selection and NDFC tests demonstrates this. These tests suggest that a close tie exists between the NDF and CC error correlation value, S. The unit selection tests suggest that a search based on S forms a more restrictive search space than an NDF based search but it also picks points which maximize the NDF. The NDFC tests showed that CC also maximizes the NDF. The NDFC tests also showed that NDFC can find smaller networks than CC. The criteria derivative tests demonstrate that for a MLP, NDF and S share common zero derivative points so that their searches may lead to the same weights being selected.

Although NDFC and NDF MLP performed well for some problems, the NDF is a costly calculation compared to the SSE and the error correlation calculations. The NDF does not appear to have a straight forward derivative calculation like SSE and error correlation. Run-time trade-offs need to be examined. It may be fruitful to use the NDF cost initially or off and on to aid the learning process. Also, the anomaly in the NDF calculation needs further examination to determine the cause and possibly a solution.

The NDF calculation needs further research. The following questions need to be addressed:

- In cascade network building, does maximizing error correlation theoretically imply maximizing the NDF?
- Can the NDF be applied initially or intermittedly to aid learning and/or architecture selection?
- Can the NDF be used to create another discriminant function which is less cost intensive?
- Do NDF related characteristics give early indication that training is not being successful?
- In NDFC, can an acceptable candidate unit output variance lower limit be estimated?
- In CC, can back-track searches be applied to find smaller networks and can heuristics be found to speed the training time.

6 Conclusions

The utility of the NDF as a criteria useful for architecture selection has been established. The NDF has successfully been used to select units for functional link networks including cascade type networks. The NDF has some although limited use in increasing the odds of finding good solutions in MLPs. Trade-offs still need to be performed to establish break-points in the usefulness of using the NDF calculation versus others which have cheaper calculations. Questions for further research have been identified.

7 References

[1] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, 1990.

[2] D. Wicker, "The Application of Artificial Neural Networks to Pattern Recognition," CEG 895, Fall 1991.

[3] R.P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP, April 1987.

[4] Andrew R. Webb and David Lowe, "The Optimised Internal Representation of Multilayer Classifier Networks Performs Nonlinear Discriminant Analysis," Neural Networks, Vol. 3, pp. 367-375, 1990.

[5] Dennis Ruck, Steven Rodgers, Matthew Kabrisky, Mark Oxley, and Bruce Suter, "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," IEEE TNN, Vol. 1, No. 4, Dec 1990.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, Vol. 2, pp 359-366, 1989.

[7] P. Gallinari, S. Thiria, F. Badran, and F. Fogelman-Soulie, "On the Relations Between Discriminant Analysis and Multilayer Perceptrons," Neural Networks, Vol. 4, pp. 349-360, 1991.

[8] David Lowe and Andrew R. Webb, "Optimised Feature Extraction and the Bayes Decision in Feed-Forward Classifier Networks," IEEE PAMI, Vol. 13, No. 4, April, 1991.

[9] Pierre Baldi and Kurt Hornik, "Neural Networks and Principal Component Analysis: Learning form Examples Without Local Minima," Neural Networks, Vol. 2, pp. 53-58, 1989.

[10] Hideki Asoh and Nobuyuki Otsu, "Nonlinear Data Analysis and Multilayer Perceptrons," IJCNN 1989, Vol. 2, pp. 411-415.

[11] Hideki Asoh and Nobuyuki Otsu, "An Approximation of Nonlinear Discriminant Analysis by Multilayer Neural Networks," IJCNN 1990, Vol. 3, pp. 211-216.

[12] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.

[13] C.W. Therrien, *Decision Estimation and Classification* Wiley, 1989.

[14] Scott E. Fahlman and Christian Lebiere, "The Cascade-Correlation Learning Architecture," CMU-CS-90-100, Feb 14, 1990.

[15] Yoh-Han Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, 1989.

[16] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in C*, Cambridge, 1989.

[17] R.A. Fisher, "The use of multiple measurements in taxonomic problems," Annunal Eugenics, 7, Part II, 179-188 (1936); also in *Contributions to Mathematical Statistics*, John Wiley, NY, 1950.

[18] P. M. Murphy and D. W. Aha, *UCI Repositor of machine learning databases* [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science.

U.S. GOVERNMENT PRINTING OFFICE 750 112